(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-5650
(P2004-5650A)
(43)公開日　平成16年1月8日(2004.1.8)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 9/45	G06F	9/44	320C	5B076
G06F 9/44	G06F	9/44	530P	5B081
	G06F	9/44	322F	
	G06F	9/06	620A	

審査請求 未請求　請求項の数 11　OL　（全 12 頁）

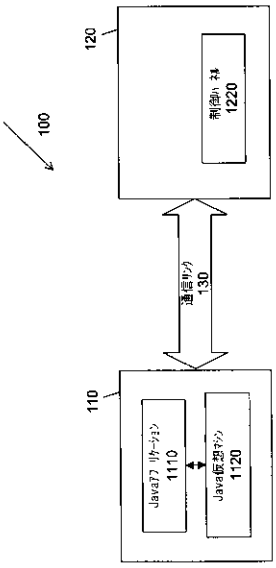
(21)出願番号　特願2003-133970(P2003-133970)
(22)出願日　平成15年5月13日(2003.5.13)
(31)優先権主張番号　10/159,528
(32)優先日　平成14年5月30日(2002.5.30)
(33)優先権主張国　米国(US)

(71)出願人　503003854
ヒューレット－パッカード　デベロップメント　カンパニー　エル．ピー．
アメリカ合衆国　テキサス州　７７０７０　ヒューストン　２０５５５　ステイト　ハイウェイ　２４９
(74)代理人　100081721
弁理士　岡田　次生
(74)代理人　100105393
弁理士　伏見　直哉
(74)代理人　100111969
弁理士　平野　ゆかり

最終頁に続く

(54)【発明の名称】実行中のコンピュータプログラムの性能を向上させる方法

(57)【要約】
【課題】コンピュータプログラムの性能向上を解決するためには非常にコストがかかる。
【解決手段】あらゆる実施形態において、プログラムの性能を向上させる技法を提供する。一実施形態では、プログラムは、Ｊａｖａ言語で書かれ、コンフィギュレーションを有するＪａｖａ仮想マシン（ＪＶＭ）を含むＪａｖａ実行環境（ＪＲＥ）で実行する。ＪＶＭにアクセスすることができる制御パネルが呼出される。制御パネルを介して、ユーザは、Ｊａｖａプログラムの実行を観察し、実行の結果を分析し、ＪＶＭのコンフィギュレーションを変更し、実行中のプログラムのクリティカル部分の再最適化を強制する。プログラム実行を観察し、結果を分析し、Ｊａｖａマシンのコンフィギュレーションを変更し、プログラムのクリティカル部分の再最適化を強制する上記ステップを繰返すことにより、ユーザは、プログラムの性能全体を向上させる。
【選択図】図１



【特許請求の範囲】
【請求項１】
コンピュータプログラムの性能を向上させる方法であって、
前記プログラムを実行し、それにより、該プログラムの性能に影響を与える、値を有するパラメータを含む仮想マシンを実行するステップと、
前記プログラム実行、前記仮想マシンおよび前記パラメータにアクセスすることができる制御パネルを呼出すステップと、
前記制御パネルを介し、前記プログラムおよび前記仮想マシンが実行している間に、
前記パラメータの前記値を新たな値になるように調整するステップと、
前記新たな値を有する前記パラメータにより前記プログラムを実行するステップと、
前記実行するステップの結果を分析するステップと、
前記分析するステップの結果に基づいてアクションを取るステップと、
を含む方法。
【請求項２】
前記プログラムは、Ｊａｖａ言語で書かれたものであり、前記仮想マシンはＪａｖａ仮想マシンである請求項１記載の方法。
【請求項３】
前記アクションを取るステップは、前記調整するステップと、前記実行するステップと、前記分析するステップと、前記アクションを取るステップとを繰返すことを含む請求項１記載の方法。
【請求項４】
前記アクションを取るステップは、前記プログラムのメソッドのコンパイルを強制することと、該プログラムのメソッドのオンスタックリプレースメント（ｏｎ－ｓｔａｃｋ　ｒｅｐｌａｃｅｍｅｎｔ）を強制することと、該プログラムを実行する前記コンピュータのメモリヒープを再構成することと、該コンピュータのメモリに対しガーベジコレクションを実行することと、前記仮想マシンを再構成することとのうちの１つまたは組合せから選択される請求項１記載の方法。
【請求項５】
前記パラメータの前記値は、前記制御パネルと前記仮想マシンとがアクセス可能な共有メモリに格納される請求項１記載の方法。
【請求項６】
前記調整するステップは、プロセス間通信（ＩｎｔｅｒＰｒｏｃｅｓｓ　Ｃｏｍｍｕｎｉｃａｔｉｏｎ）プロトコルを使用する請求項１記載の方法。
【請求項７】
前記パラメータは、前記仮想マシンのコンフィギュレーションの一部である請求項１記載の方法。
【請求項８】
ネットワークを介して、前記プログラムを実行している前記コンピュータに接続された表示装置に、前記制御パネルを表示するステップをさらに含む請求項１記載の方法。
【請求項９】
後の使用のために前記新たな値を保存するステップをさらに含む請求項１記載の方法。
【請求項１０】
前記仮想マシンのトレース機能を起動するステップをさらに含む請求項１記載の方法。
【請求項１１】
仮想マシンをチューニングする方法であって、
プログラムを実行し、それにより、該プログラムの性能に影響を与えるコンフィギュレーションを有する前記仮想マシンを呼出すステップと、
前記プログラム実行および前記仮想マシンの前記コンフィギュレーションにアクセスすることができる制御パネルを呼出すステップと、
前記制御パネルを介し、前記プログラムおよび前記仮想マシンが実行している間に、

前記プログラムの実行からもたらされるデータを分析するステップと、
前記分析するステップの結果に基づいて前記仮想マシンの前記コンフィギュレーションを調整するステップと、
前記プログラムを前記仮想マシンの前記調整されたコンフィギュレーションで実行するステップと、
を含み、
前記仮想マシンは、前記プログラムのプログラムコードを実行する環境を提供する方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、概してコンピュータプログラムに関し、特に実行中のかかるプログラムの性能を向上させることに関する。
【０００２】
【従来の技術】
最初許容可能に実行中のアプリケーションプログラムは、長い間実行された後、プログラムの作業負荷特性の変化を含むあらゆる理由により、性能が低下する可能性がある。たとえば、Ｊａｖａ実行環境（Ｊａｖａ　Ｒｕｎｔｉｍｅ　Ｅｎｖｉｒｏｎｍｅｎｔ（ＪＲＥ））において、Ｊａｖａ仮想マシン（Ｊａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅ（ＪＶＭ））は、Ｊａｖａアプリケーションの実行速度を上げるために動的最適化コンパイラに頼る。アプリケーションが実行している間、ＪＶＭは、最も頻繁に呼出されるメソッド、すなわち「ホット（ｈｏｔ）」メソッドを識別し、それらを最適化し、最適化したメソッドをコードキャッシュに配置することにより、これらのメソッドの後続する呼出しが、それらメソッドのキャッシュバージョンを使用することでより効率的になるようにする。コードキャッシュは、ホットメソッドの最適化コードを格納するメモリ内の指定領域である。ホットメソッドのセットは、通常、プログラムが実行する特定のタイプの負荷かまたは動作のセットに関連付けられる。２つの異なる負荷を実行しているプログラムは、ホットメソッドの２つの異なるセットを提供する。たとえば、税金プログラムは、個人に対して税金を決める場合、法人に対して税金を決める同じ税金プログラムに関連するホットメソッドのセットとは異なるホットメソッドのセットに関連する。それは、個人に対して税金を決めるメソッドは、法人に対して税金を決めるメソッドとは異なるためである。したがって、ホットメソッドは、通常、プログラム負荷が変化した場合に識別される。さらにホットメソッドが識別され最適化されるにしたがい、アプリケーションはより効率的に実行する。しかしながら、アプリケーションがある期間実行した後、ホットメソッドは安定状態に達し、すなわち、いかなる新たなホットメソッドも識別することができなくなる。アプリケーション性能は、安定状態メソッドがいかに適切に最適化されるかによって決まる。
【０００３】
アプリケーション最適化のレベルおよび範囲に影響を与える閾値は、通常、Ｊａｖａ環境が顧客に出荷される前に事前設定される。積極的な（ａｇｇｒｅｓｓｉｖｅ）閾値も保守的な（ｃｏｎｓｅｒｖａｔｉｖｅ）閾値もともに、性能を低下させる可能性がある。たとえば、コンパイル中にインライン化すべきメソッドのコードの量が多い積極的な閾値は、過度の命令キャッシュ（Ｉ－ｃａｃｈｅ）ミスをもたらす可能性があり、保守的な閾値は、コードの大部分を最適化されないままにする。両方の状況により、アプリケーション性能が低下する。
【０００４】
一般に、ユーザは、性能が低下しているアプリケーションを認識すると、そのアプリケーションを検査し、アプリケーション性能に影響を与えるＪＶＭのパラメータを識別し、これらのパラメータに対してより適切な値を再割当てする。しかしながら、新たな変化が有効であるために、変更されたパラメータに基づく新たな環境をインストールしなければならず、このためには既存のアプリケーションを停止させる必要がある。アプリケーション

が停止されている間、そのアプリケーションを使用することができず、業務の混乱がもたらされる。あらゆる状況において、問題を補正するために、ユーザはＪａｖａ環境を提供する事業機関と連係する必要がある。これらの状況において、ユーザは、その機関の現場に物理的に出向かなければならず、そこで、アプリケーションのソフトウェアおよびハードウェア環境と性能低下をもたらしている状況とを再現する必要があり、それらはすべて非常にコストがかかる可能性がある。多くの状況では、性能低下をもたらす状況を作り出すことは容易ではない。さらに、アプリケーションがデバグされた後、一般的に新たなＪａｖａ環境は変更され、その後、ユーザは、ユーザのマシンにおける変更を含めて新たなＪａｖａ環境をインストールしなければならない。これには、また、アプリケーションを停止させる必要があり、上述したような混乱をもたらす。

【０００５】

【発明が解決しようとする課題】

上述したことに基づき、上記欠点および関連する問題を解決するメカニズムへのニーズが高まっている。

【０００６】

【課題を解決するための手段】

本発明は、あらゆる実施形態において、プログラムの性能を向上させる技法を提供する。一実施形態では、プログラムは、Ｊａｖａ言語で書かれ、コンフィギュレーションを有するＪａｖａ仮想マシン（ＪＶＭ）を含むＪａｖａ実行環境（ＪＲＥ）で実行する。ＪＶＭにアクセスすることができる制御パネルが呼出される。制御パネルを介して、ユーザは、Ｊａｖａプログラムの実行を観察し、実行の結果を分析し、ＪＶＭのコンフィギュレーションを変更し、実行中のプログラムのクリティカル部分の再最適化を強制する。プログラム実行を観察し、結果を分析し、Ｊａｖａマシンのコンフィギュレーションを変更し、プログラムのクリティカル部分の再最適化を強制する上記ステップを繰返すことにより、ユーザは、プログラムの性能全体を向上させる。

【０００７】

本発明を、添付図面の各図において実施例として例示する。図面において、同じ参照符号は同様の要素を参照する。

【０００８】

【発明の実施の形態】

以下の説明では、説明の目的で、本発明の完全な理解を提供するために多くの具体的な詳細を示す。しかしながら、当業者には、本発明がこれらの具体的な詳細無しに実施されてよい、ということが明らかとなろう。すなわち、周知の構造および装置は、発明を不明確にするのを回避するためにブロック図形態で示す。

【０００９】

図１は、本発明の実施形態を実施することができるシステム１００を示す。システム１００は、第１のコンピュータ１１０と第２のコンピュータ１２０とを含み、それらを、例示の目的のために、それぞれアプリケーションサーバ１１０と性能デバッギングマシン１２０と呼ぶ。サーバ１１０とマシン１２０とは、通信リンク１３０を介して接続される。

【００１０】

サーバ１１０は、あらゆるアプリケーションプログラムを実行する。それらのうちの１つを、プログラム１１１０として示す。一実施形態では、プログラム１１１０は、Ｊａｖａ仮想マシン（ＪＶＭ）１１２０を含むＪａｖａ実行環境（ＪＲＥ）内で作動するＪａｖａプログラムである。通常、プログラム１１１０は、ＪＶＭ１１２０の制御下で最適化され実行される。そして、プログラム１１１０は、通常通信リンク１３０、インターネット等のネットワークを介してユーザにサービスを提供する。これらサービスには、たとえば、ウェブ、データベース、電子メール、セキュリティ、通信等が含まれる。

【００１１】

Ｊａｖａ仮想マシン１１２０は、ＪＶＭ１１２０が動作するコンピュータプラットフォームから独立した、Ｊａｖａバイトコードを実行する環境を提供する。概して、ＪＶＭ１１

２０は、ハードウェアで実行しているソフトウェアとサーバ１１０のオペレーティングシステムとで実現される。このため、ＪＶＭ１１２０は、バイトコードの形式での汎用プログラム表現がサーバ１１０で実行されるのを可能にする環境を提供する。ＪＶＭ１１２０は、Ｊａｖａプログラムを最適化しＪａｖａバイトコードをサーバ１１０によって実行される動作に翻訳する責任を有する。ＪＶＭ１１２０は、アプリケーションプログラム、たとえばプログラム１１１０の性能に影響を与えるあらゆるパラメータを含む。

【００１２】

マシン１２０は、制御パネル１２２０を表示することができるモニタまたは画面を含む。この図１の実施例では、制御パネル１２２０はサーバ１１０からリモートであり、すなわち、制御パネル１２２０は通信リンク１３０を介してサーバ１１０と通信する。しかしながら、制御パネル１２２０は、ローカルであってよく、たとえば、サーバ１１０に対してローカルであるモニタに表示されてよい。一実施形態では、制御パネル１２２０は、ユーザがＪＶＭ１１２０にアクセスしそれを変更するのを可能にする、ＪＶＭ１１２０から呼出されるソフトウェアパッケージである。さらに、Ｊａｖａ環境を提供している機関に関連する顧客サポート技術者は、グラフィカルユーザインタフェース（ＧＵＩ）の形態の制御パネル１２２０を介して、リモートにプログラム１１１０の性能を分析し向上させる。

【００１３】

通信リンク１３０は、サーバ１１０がマシン１２０と通信するためのメカニズムである。通信リンク１３０は、伝送制御プロトコル／インターネットプロトコル（Ｔｒａｎｓｍｉｓｓｉｏｎ Ｃｏｎｔｒｏｌ Ｐｒｏｔｏｃｏｌ／Ｉｎｔｅｒｎｅｔ Ｐｒｏｔｏｃｏｌ（ＴＣＰ／ＩＰ））、公衆交換電話網（Ｐｕｂｌｉｃ Ｓｗｉｔｃｈｅｄ Ｔｅｌｅｐｈｏｎｅ Ｎｅｔｗｏｒｋ（ＰＳＴＮ））、デジタル加入者線（Ｄｉｇｉｔａｌ Ｓｕｂｓｃｒｉｂｅｒ Ｌｉｎｅ（ＤＳＬ））、ケーブルネットワーク、衛星対応、無線対応等の通信プロトコルのうちの１つまたは組合せを利用する単一ネットワークかまたはネットワークの組合せであってよい。一実施形態では、通信リンク１３０はインターネットである。

【００１４】

Ｊａｖａ仮想マシンのコンフィギュレーション

Ｊａｖａ仮想マシン１１２０は、あらゆる機能を実行し、あらゆるパラメータを含み、それらはともにプログラム１１１０の性能に影響を与える。たとえば、ＪＶＭ１１２０は、実行中のＪａｖａプログラムがメモリを使い尽さないようにサーバ１１０のメモリのデータを管理する、ガーベジコレクション（ｇａｒｂａｇｅ ｃｏｌｌｅｃｔｉｏｎ）を実行する。メモリの管理は時間と資源とを費やし、そのためプログラム１１１０の性能に影響を与える可能性がある。ガーベジコレクションの頻度を変更することにより、プログラム１１１０の性能が変化する。一実施形態では、ＪＶＭ１１２０は、ガーベジコレクションがどれくらいの頻度で実行されるべきかを示す閾値と、メモリブロックに対しガーベジコレクションが実行される優先度を示すメカニズムと、を含む。ヒープは、ガーベジコレクションが行われるオブジェクトが配置されるメモリロケーションである。ヒープが充填されると、ガーベジコレクタが始動する。ヒープが小さいほど、ガーベジコレクションが頻繁に発生し、そのためシステム性能が劣化する可能性がある。ＪＶＭ１１２０は、ヒープサイズを調整するパラメータを含む。

【００１５】

「ホットな」Ｊａｖａプログラムメソッドは、極めてしばしば呼出され、インタプリタ型ではなくコンパイル型で実行される。それは、コンパイル型メソッドの実行はインタプリタ型メソッドの実行より高速であるためである。一実施形態では、ＪＶＭ１１２０は、メソッドがホットであるか否かを判断する閾値を含む。たとえば、５０００という閾値は、メソッドが少なくとも５０００回呼出されるまではホットメソッドとみなされないことを示す。同様に、１００００という閾値は、メソッドが少なくとも１００００回呼出されるまではホットメソッドとみなされないことを示す。ホットメソッドはしばしばプログラムにおいて呼出されるため、ホットメソッドのセットは、プログラムのクリティカル部分と

いえる。
【００１６】
Ｊａｖａメソッドを、異なるレベルの積極性（ａｇｇｒｅｓｓｉｖｅｎｅｓｓ）でコンパイルすることができる。積極的にコンパイルされるメソッドほど、それほど積極的にコンパイルされないメソッドより最適化のために考慮されるコードの範囲または領域が広く、そのためコンパイルするコードが多い可能性がある。この結果、多くのコードを有する、より積極的にコンパイルされるメソッドが、コードの少ないメソッドと同程度の頻度ではキャッシュに格納することができないため、コンパイルにより長い時間がかかり、かつ／または命令キャッシュミス率が高くなる可能性がある。一実施形態では、ＪＶＭ１１２０は、ユーザがメソッドをコンパイルする積極性を設定するためのあらゆるフラグを含む。また、ＪＶＭ１１２０は、要求時にメソッドが強制的にコンパイルされるようにするメカニズムも可能とする。一実施形態では、コンパイラは「最適化コンパイラ」であるため、再コンパイルを強制することは、最適化を強制することを意味してよく、何をおよび／またはどの程度最適化するかは、最適化コンパイラによって使用されるパラメータおよび／または閾値に基づく。さらに、ホットメソッドはプログラムのクリティカル部分と言えるため、ホットメソッドの再コンパイルを強制することは、プログラムの最適化を強制することであると言える。
【００１７】
また、ＪＶＭ１１２０は、実行中のプログラムの一部を通常その部分かまたはメソッドのよりよく最適化されたバージョンよって置換することを言う、オンスタックリプレースメント（ｏｎ－ｓｔａｃｋ　ｒｅｐｌａｃｅｍｅｎｔ）を強制するメカニズムも含む。
【００１８】
ＪＶＭ１１２０は、最初にサーバ１１０にインストールされた時、パラメータのセットに合せて構成され、そのため何らかの所定の機能を実行することになっている。しかしながら、ユーザは、たとえば制御パネル１２２０を介して、アプリケーションプログラム１１１０およびＪＶＭ１１２０が実行している間にこれらパラメータを調整することができる。
【００１９】
Ｊａｖａ仮想マシンのコンフィギュレーションの変更
本発明の実施形態により、ユーザは、ＪＶＭ１１２０のコンフィギュレーションをリアルタイムに、すなわち「オンザフライ（ｏｎ　ｔｈｅ　ｆｌｙ）」で（これは通常、プログラム１１１０が実行中であるまたは実行されているという事実を言う）変更することができる。ＪＶＭ１１２０のコンフィギュレーションを変更することは、あらゆるパラメータおよび閾値を調整することを含み、アプリケーション１１１０および／またはＪＶＭ１１２０のチューニング、デバッギングおよび／または性能の向上が望ましい場合に適用可能である。例示の目的のために、この文書では、性能の向上という用語を使用する。しかしながら、例示的な概念は、向上、チューニング、デバッギングおよびそれらの等価に適用可能である。ＪＶＭ１１２０のチューニングは、一般に、新しい機能がアプリケーション１１１０に追加される、アプリケーション１１１０の負荷が変化する、ＪＶＭ１１２０の新しいバージョンが導入される、ＪＶＭ１１２０により新しいアプリケーションが最初に実行される等の場合に発生する。
【００２０】
制御パネル１２２０を介して、ユーザは、たとえばＪＶＭ１１２０内部のトレース特性をその「オン」状態に設定することにより、ＪＶＭ１１２０のトレースを起動することができる。トレースを通して、ユーザは、性能の低下の原因を識別するために有用な情報を獲得する。そして、ユーザは、獲得した情報を分析し、必要に応じてＪＶＭ１１２０のコンフィギュレーションを変更する。ＪＶＭ１１２０のコンフィギュレーションを変更することは、ＪＶＭ１１２０が実行するあらゆる動作の頻度および持続時間を制御するか、またはプログラムのいずれの部分をＪＶＭ１１２０の最適化コンパイラを通して処理するべきか等の、プログラム１１１０の最適化レベルに直接影響を与える、あらゆる閾値を調整す

ることを含む。たとえば、ユーザは、ガーベジコレクションの頻度を変更して新たな頻度を即座に起動する、コードキャッシュのコードを再生成する、等行ってよい。

【００２１】

パラメータおよび実施態様次第で、ＪＶＭ１１２０の各パラメータは、制御パネル１２２０に表示されるオン／オフボタン、スライディングバー、選択リスト等の制御メカニズムに対応する。ユーザは、対応する制御メカニズムを調整することによりパラメータを調整する。たとえば、強制されたコンパイルまたは強制されたオンスタックリプレースメントの場合、メソッドはオン／オフボタンに対応し、このボタンをクリックすることにより、対応する強制されたコンパイルまたは強制されたオンスタックリプレースメントが可能になりまたは不能になる。別の例では、ガーベジコレクションをトリガする閾値か、またはメソッドがホットであるものと示す閾値等の閾値を設定するパラメータが、閾値の範囲に対応するスライディングバーによって表現される。スライディングバーを調整することにより、対応する閾値が調整される。

【００２２】

一実施形態では、プログラム１１１０が実行している間に調整可能であることが望ましいパラメータは、サーバ１１０の共有メモリに格納される。このメモリは、複数のプロセスによって変更することができるメモリの領域を言う。制御パネル１２２０とＪＶＭ１１２０とはともに、この共有メモリにアクセスすることができる２つのプロセスであるとみなされる。プロセスとしての制御パネル１２２０を介して、ユーザは、共有メモリのパラメータ値を変更し、それにより、アプリケーションプログラム１１１０が新たに変更されたパラメータによって即座に実行されることを可能にし、その結果アプリケーション１１１０の性能がリアルタイムに向上する。代替実施形態では、パラメータはＪＶＭ１１２０のデータ空間に格納され、制御パネル１２２０およびＪＶＭ１１２０は２つのプロセスのままである。そして、ユーザは、プロセス間通信（ＩｎｔｅｒＰｒｏｃｅｓｓ Ｃｏｍｍｕｎｉｃａｔｉｏｎ（ＩＰＣ））プロトコルを使用して、ＪＶＭ１２２０のデータ空間のパラメータを直接調整する。代替実施形態では、制御パネル１１２０はＪＶＭ１１２０の一部であり、ユーザは制御パネル１２２０を介してＪＶＭ１１２０のデータ空間からのパラメータを変更する。プロセスは、実行プログラムまたはタスクとして定義することができ、それは、プログラムとオペレーティングシステムによって使用される情報を保持するブックとの組合せを言う。タスクは、プログラムをタスク番号で識別する。プロセスは、プロセスの他の部分から独立して実行することができるスレッド内に生成することができる。

【００２３】

ユーザは、制御パネル１２２０を介して、プログラム１１１０の性能をリアルタイムに観察し、かかる性能を向上させる適当な動作を行うことができる。たとえば、ユーザは、プログラム１１１０およびＪＶＭ１１２０がいかに実行しているかをユーザが見るのを可能にするＪＶＭ１１２０のトレース機能を起動することにより、プログラム１１１０の何のメソッドがインタプリタ型ではなくコンパイル型になるか、キャッシュミス率は何か、あらゆるパラメータの閾値がいかなる値に設定されるか、等を識別する。観察とリアルタイムに獲得されるデータとに基づいて、ユーザは、既存のデータを分析する、ＪＶＭ１１２０の新たなデータの各種セットと新たなコンフィギュレーションとを有するプログラム１１１０を実行することにより実験を行う等により、問題を解決する。望ましい場合、ユーザは、利用可能な性能ツールを使用することにより、データ、プログラム１１１０、ＪＶＭ１１２０等を分析する。性能分析ツールの例には、カリフォルニア州、Ｓａｎｔａ Ｃｌａｒａのインテル（Ｉｎｔｅｌ）によるＰｒｏｓｐｅｃｔ、Ｇｐｒｏｆ、Ｖｔｕｎｅ、カリフォルニア州、Ｐａｌｏ Ａｌｔｏのヒューレット・パッカード・カンパニー（Ｈｅｗｌｅｔｔ－Ｐａｃｋａｒｄ Ｃｏｍｐａｎｙ）によるＨＰＪｍｅｔｅｒおよびＧｌａｎｃｅ等が含まれる。たとえば、ユーザは、メソッドが極めてしばしば呼出されるが、その対応するホットメソッド閾値は非常に高い値に設定されている、ということを観察することにより、この値を低下させることによって、このメソッドがホットメソッドとみなされ

それによりインタプリタ型ではなくコンパイル型で実行されるようにする。対照的に、メソッドが非常に積極的にコンパイルされているために非常に多くのキャッシュミスをもたらす場合、ユーザは、そのメソッドをより積極的でないオプションで再コンパイルする。デバッギングプロセスを支援するために、ユーザは、メソッドＡのコンパイルを強制し、かつ／またはメソッドＢのオンスタックリプレースメントを強制する等を行う。

【００２４】

一実施形態では、各デバッギングセッション後に、ＪＶＭ１１２０のコンフィギュレーションテンプレートが生成され、たとえばＪＶＭ１１２０のあらゆるパラメータの新しい値を保持するファイルに格納される。このように、各テンプレートは、通常、アプリケーション１１１０がデバグされ実行される、特定のかつ明確なプログラム負荷および環境に関連付けられる。コンフィギュレーションテンプレートは、顧客に対し、ＪＶＭ１１２０を再構成して、同様の状況で使用できるようにする迅速な方法を提供する。たとえば、アプリケーション１１１０がデータベース要求を高負荷で実行している場合、テンプレートＴ１が生成され、アプリケーション１１１０がウェブサービスを実行している場合、テンプレートＴ２が生成され、等である。データベースまたはウェブサービスを実行しているアプリケーションが発生すると、ユーザは、迅速にテンプレートＴ１またはＴ２をそれぞれロードすることができる。

【００２５】

プログラム性能を向上させる方法を示すステップ

図２は、一実施形態によるプログラム性能を向上させる方法を示すフローチャートである。

【００２６】

ステップ２０４において、ユーザは、制御パネル１２２０を使用してアプリケーション１１１０およびＪａｖａ仮想マシン１１２０にアクセスする。

【００２７】

ステップ２０８において、ユーザは、アプリケーション１１１０の性能を検査する。

【００２８】

ステップ２１２において、ユーザは、アプリケーション１１１０の性能が満足のいくものであるか否かを判断する。性能が不満足なものである場合、ユーザは、ステップ２１６において、ＪＶＭ１１２０のトレースを起動する。

【００２９】

ステップ２２０において、ユーザはトレースされたデータを分析する。

【００３０】

ステップ２２４において、ユーザは、たとえば各種閾値および／またはパラメータを変更することにより、ＪＶＭ１１２０を再構成する。

【００３１】

ステップ２２８において、ユーザは、必要な場合、ホットメソッドの再コンパイルを強制し、かつ／またはいくつかの目下実行中のメソッドのオンスタックリプレースメントを強制する、等を行う。

【００３２】

そして、ユーザは、ステップ２１２においてアプリケーション１１１０の性能が満足のいくものであると判断するまで、上記ステップ２０８、２１２、２１６、２２０、２２４および２２８を繰返す。性能が満足のいくものである場合、ユーザは、ステップ２３２において、ＪＶＭ１１２０のコンフィギュレーションが変更されたか否かを判断し、変更された場合、ユーザは、ステップ２３６において、新たなコンフィギュレーションを保存する。本方法は、ステップ２４０において完了する。しかしながら、ＪＶＭ１１２０のコンフィギュレーションが変更されなかった場合、ユーザは、ステップ２３６におけるコンフィギュレーションの保存のステップをスキップし、方法はステップ２４０で完了する。

【００３３】

本発明の実施形態はいくつかの利点を提供する。例示の目的のために、チューニングとい

う用語を使用するが、これら利点は、性能の向上、パラメータの調整等、他の同様な状況において適用可能である。アプリケーション１１１０のチューニングを、リアルタイムに、アプリケーション１１１０が実行している場所からリモートに、実負荷下で実行することができる。別のチューニング環境を構築する必要はない。チューニングが単純で容易で安価となるため、チューニングをより頻繁に実行することができる。

【００３４】

コンピュータシステム概観

図３は、本発明の一実施形態を実施することができるコンピュータシステム３００を示すブロック図である。たとえば、コンピュータシステム３００を、サーバ１１０またはデバッギングマシン１２０等として動作するようにように実施することができる。一実施形態では、コンピュータシステム３００は、すべてバス３２４に接続される、中央演算処理装置（ＣＰＵ）３０４と、ランダムアクセスメモリ（ＲＡＭ）３０８と、リードオンリメモリ（ＲＯＭ）３１２と、記憶装置３１６と、通信インタフェース３２０とを含む。

【００３５】

ＣＰＵ３０４は、ロジックを制御し、情報を処理し、コンピュータシステム３００内のアクティビティを調整する。一実施形態では、ＣＰＵ３０４は、たとえば入力装置３２８から表示装置３３２へのデータの移動を調整することにより、ＲＡＭ３０８およびＲＯＭ３１２に格納された命令を実行する。ＣＰＵ３０４は、１つまたは複数のプロセッサを含んでよい。

【００３６】

ＲＡＭ３０８は、通常メインメモリと呼ばれ、ＣＰＵ３０４によって実行すべき情報および命令を一時的に格納する。ＲＡＭ３０８内の情報は、入力装置３２８から取得されるか、またはＣＰＵ３０４によって実行される命令によって要求されるアルゴリズム的プロセスの一部としてＣＰＵ３０４によって生成されてよい。

【００３７】

ＲＯＭ３１２は、一旦ＲＯＭチップに書込まれると読取専用であり変更または除去されない、情報および命令を格納する。一実施形態では、ＲＯＭ３１２は、コンピュータシステム３００のコンフィギュレーションおよび初期動作に対するコマンドを格納する。

【００３８】

フロッピディスク、ディスクドライブまたはテープドライブ等の記憶装置３１６は、コンピュータシステム３００が使用する情報を永続的に格納する。

【００３９】

通信インタフェース３２０により、コンピュータシステム３００は他のコンピュータまたは機器とインタフェースすることができる。通信インタフェース３２０は、たとえば、モデム、サービス統合デジタル網（ＩＳＤＮ）カード、ローカルエリアネットワーク（ＬＡＮ）ポート等であってよい。当業者は、モデムまたはＩＳＤＮカードが電話回線を介してデータ通信を提供し、ＬＡＮポートがＬＡＮを介してデータ通信を提供することを認めるであろう。また、通信インタフェース３２０は、無線通信を可能としてもよい。

【００４０】

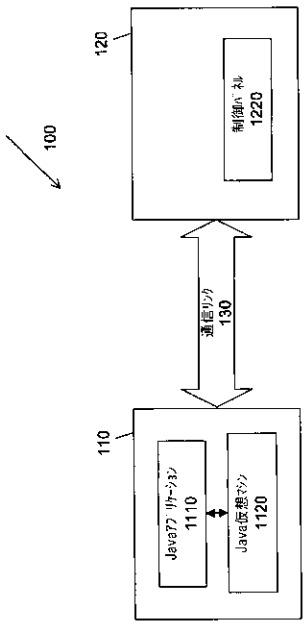
バス３２４は、コンピュータシステム３００が使用する情報を通信するいかなる通信メカニズムとすることも可能である。図３の実施例では、バス３２４は、ＣＰＵ３０４、ＲＡＭ３０８、ＲＯＭ３１２、記憶装置３１６、通信インタフェース３２０等の間でデータを転送する媒体である。

【００４１】

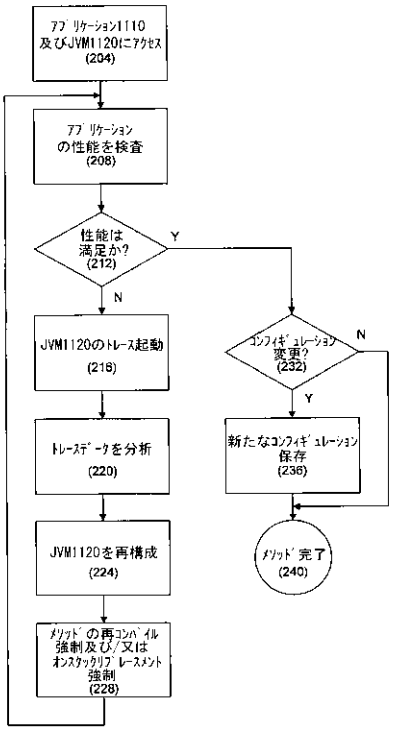
コンピュータシステム３００は、一般に、入力装置３２８と表示装置３３２とカーソル制御３３６とに結合される。英数字キーおよび他のキーを含むキーボード等の入力装置３２８は、情報およびコマンドをＣＰＵ３０４に通信する。ブラウン管（ＣＲＴ）等の表示装置３３２は、コンピュータシステム３００のユーザに対して情報を表示する。マウス、トラックボールまたはカーソル方向キー等のカーソル制御３３６は、方向情報およびコマンドをＣＰＵ３０４に通信し、表示装置３３２上でのカーソル移動を制御する。

【００４２】
コンピュータシステム３００は、１つまたは複数のネットワークを通して他のコンピュータまたは機器と通信することができる。たとえば、コンピュータシステム３００は、通信インタフェース３２０を使用することにより、ネットワーク３４０を通して、プリンタ３４８に接続された別のコンピュータ３４４と、またはワールドワイドウェブ３５２を通してサーバ３５６と通信する。ワールドワイドウェブ３５２は、一般に「インターネット」と呼ばれる。代替的に、コンピュータシステム３００は、ネットワーク３４０を介してインターネット３５２にアクセスすることができる。
【００４３】
コンピュータシステム３００を使用して、上述した技法を実施してよい。各種実施形態において、ＣＰＵ３０４は、ＲＡＭ３０８にもたらされる命令を実行することにより本技法のステップを実施する。代替実施形態では、上述した技法を実施するために、ソフトウェア命令の代りにまたはそれと組合せてハードワイヤード回路を使用してよい。したがって、本発明の実施形態は、ソフトウェア、ファームウェア、ハードウェアまたは回路のうちの任意の１つまたは組合せに限定されない。
【００４４】
ＣＰＵ３０４によって実行される命令は、１つまたは複数のコンピュータ読取可能媒体に格納され、かつ／またはそれによって達成されてよい。コンピュータ読取可能媒体とは、コンピュータがそこから情報を読出す任意の媒体を言う。コンピュータ読取可能媒体は、例えば，フロッピディスク、ハードディスク、ジップ（ｚｉｐ）ドライブカートリッジ、磁気テープまたは他の任意の磁気媒体、ＣＤ－ＲＯＭ、ＣＤ－ＲＡＭ、ＤＶＤ－ＲＯＭ、ＤＶＤ－ＲＡＭまたは他の任意の光媒体、紙テープ、パンチカードまたは孔のパターンを有する他の任意の物理媒体、ＲＡＭ、ＲＯＭ、ＥＰＲＯＭまたは他の任意のメモリチップまたはカートリッジであってよい。また、コンピュータ読取可能媒体は、同軸ケーブル、銅線、光ファイバ、音波または電磁波、容量結合または誘導結合等であってもよい。例として、ＣＰＵ３０４が実行すべき命令は、１つまたは複数のソフトウェアプログラムの形式であり、バス３２４を介してコンピュータシステム３００とインタフェースするＣＤ－ＲＯＭに最初に格納される。コンピュータシステム３００は、これらの命令をＲＡＭ３０８にロードし、いくつかの命令を実行し、いくつかの命令を通信インタフェース３２０、モデムおよび電話線を介してネットワーク、たとえばネットワーク３４０、インターネット３５２等に送信する。ネットワークケーブルを介してデータを受取るリモートコンピュータは、受取った命令を実行し、そのデータを、記憶装置３１６に格納されるようにコンピュータシステム３００に送信する。
【００４５】
上述した明細書において、本発明を、特定の実施形態を参照して説明した。しかしながら、本発明のより広い精神および範囲から逸脱することなく、あらゆる変更および変形を行ってよい、ということが明らかとなろう。したがって、明細書および図面は、限定的ではなく例示的であるとみなされなければならない。
【図面の簡単な説明】
【図１】本発明の実施形態を実施することができるシステムの図。
【図２】一実施形態によるプログラムの性能を向上させる方法を示すフローチャート。
【図３】本発明の実施形態を実施することができるコンピュータシステムを示す図。
【符号の説明】
１１１０　Ｊａｖａアプリケーション
１１２０　Ｊａｖａ仮想マシン
１３０　通信リンク
１２２０　制御パネル

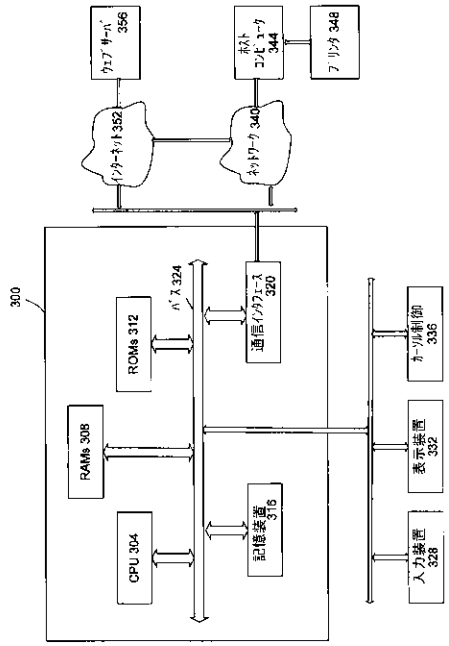
【図１】


100
120
制御パネル
1220
通信リンク
130
110
Java アプリケーション
1110
Java仮想マシン
1120

【図２】


アプリケーション1110及びJVM1120にアクセス
(204)
アプリケーションの性能を検査
(208)
性能は満足か?
(212)
Y
N
JVM1120のトレース起動
(216)
トレースデータを分析
(220)
JVM1120を再構成
(224)
メソッドの再コンパイル強制及び/又はオンスタックリプレースメント強制
(228)
コンフィギュレーション変更?
(232)
N
Y
新たなコンフィギュレーション保存
(236)
メソッド完了
(240)

【図３】


300
ウェブサーバ 356
ホストコンピュータ 344
リンク 348
インターネット 352
ネットワーク 340
バス 324
通信インタフェース 320
ROMs 312
RAMs 308
CPU 304
記憶装置 316
カーソル制御 336
表示装置 332
入力装置 328

フロントページの続き

(72)発明者　ノウバー・パータミアン
アメリカ合衆国９４０４１カリフォルニア州マウンテン・ヴュー、チキータ・アヴェニュー　３０３、ナンバー　７
(72)発明者　ローラン・モリチェッティ
アメリカ合衆国９５１２６カリフォルニア州サン・ノゼ、ザ・アラメダ　１３４６、ナンバー　４０５
(72)発明者　アミタブ・ネネ
アメリカ合衆国９５０５０カリフォルニア州サンタ・クララ、ベントン・ストリート　１０５０、ナンバー　２３０１
(72)発明者　アンドリュー・トリック
アメリカ合衆国９５０１４カリフォルニア州クパーティノ、セダーブロック・テラス　２０５３７

Ｆターム(参考) 5B076 DA01 DC00 EA13 EA17
5B081 AA09 BB08 CC21 DD01